Cardiff University

School of Computer Science and Informatics

Toxic Comment Classification

MSc in Artificial Intelligence

Author: Nisargkumar Rajvi (2097941)

Supervisor: Proff. Irena Spasic

Moderator: Proff. Bailing Dang

Abstract

Two of the most in-demand fields for computer programmers are supervised learning and text classification. There have been significant advancements in Natural Language Processing over the past two decades. The proliferation of textual information available online has increased at an exponential rate since the advent of digitalization. When these text data need to be sorted into categories that have already been established, supervised classification comes into play. Labelled data is used as input for a Machine Learning algorithm, which then generates models to classify data that has not yet been seen.

This work compiles the findings of several investigations and field notes about the use of multiple labels for classifying user feedback. In this study, I sought to manage a comment-classification Kaggle competition. Using a tagged dataset from the Wikipedia Talk Page, I trained a Recurrent Neural Network model using a Bidirectional Long Short-term Memory activation function to classify harmful comments. For each model, I've employed a unique pre-processing technique. I have also analysed the class imbalance concerns related with the dataset.

This thesis requires, among other things, the compilation of a corpus of examples of toxic, abusive, hateful language and associated concepts. Sadly, this is mostly a manual process, and as such, it takes a long time. A web-based application is developed to shorten the time needed to generate this and subsequent data sets.

Disclaimer: The dataset includes text that some people may find objectionable, profane, vulgar and offensive.

Acknowledgments

I'd want to express my gratitude to the following persons, without whom I never would have finished my dissertation.

Proff. Irena Spasic, my advisor, whose expertise and understanding guided me through my project and who encouraged me when times were tough. The appreciation is genuine, Prof. Irena.

Thank you very much to the kind people at the NHS who helped me while I was unwell during my dissertation research.

Furthermore, I'd want to express my deepest gratitude to my loved ones for their unwavering encouragement and assistance throughout this study.

Contents

List of Tables

List of Figures

Table of Abbreviations

EDA – exploratory data analysis

TF-IDF – Term Frequency-Inverse Document Frequency

UI – User Interface

CNN - Convolutional Neural Networks

RNN - Recurrent Neural Network

LSTM – Long-short term memory

NLP - Natural Language Processing

CV - Computer Vision

biLSTM – Bidirectional Long-short term memory

NB-SVM - Naïve Bayes Support Vector Machine

AI – Artificial Intelligence

ML – Machine Learning

GPU - Graphic Processing Unit

Introduction

1.1 Research Motivation

The internet is a public discussion forum where everyone may openly share their views. Harassment and abuse, on the other hand, discourage individuals from sharing their thoughts and disrupt the internet ecosystem. Because platforms fail to properly encourage dialogues, many communities limit or altogether disable user comments if they are hostile.

Some users take advantage of the open forum provided by social media sites to spread hate speech and other forms of harassment. Adults may be able to cope with this threat, but young people, especially adolescents, are at a far higher risk of developing severe mental health problems (Adadi, A & Berrada, M., 2018). Since the Covid-19 shutdown, there has been a 70% spike in bullying and hate speech among teens and children (Maya Shwayder & Mythili Sampathkumar , 2020). The global initiative to eliminate toxic content has contributed to a growing interest in social and ethical concerns within the artificial intelligence and natural language processing communities.

Much of the prior work in this field has been on developing effective categorisation systems for determining whether or not the full material is harmful. Models in this category include some of the most common types of machine learning algorithms, including logistic regression and support vector machines (Hovy, D & Waseem, Z., 2016) including convolutional neural networks, recurrent neural networks, and attention networks (Gupta, K & Muresan, S., 2019).

Jigsaw, an Alphabet Inc. subsidiary "dedicated to understanding global challenges and applying technological solutions, from countering extremism, online censorship, and cyber-attacks, to protecting access to information," is hosting a competition in which participants are asked to analyse Wikipedia comments and detect various types of toxicity. The dataset includes over 150,000 user comments on Wikipedia that were manually categorised into two categories based on whether or not they contained toxic language, severe toxic language, obscenities, threats, insults, or hate speech.

One of the most powerful methods for extracting, analysing, and categorising key aspects from textual data is NLP with Neural Networks. In this dissertation, I will use a multi-label text dataset to conduct text classification using pre-processing techniques and a LSTM, in order to identify various types of online toxicity

1.2 Aims and Objectives

The aim of this project is to identify and categorise toxic internet comments provided in the dataset (https://www.kaggle.com/datasets/julian3833/jigsaw-toxic-comment-classification-challenge?datasetId=1709138) , and the dataset used for this project is the one supplied by jigsaw for the Toxic comment classification challenge (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge) on Kaggle.

This project's overarching goal is to develop a multi-faceted model that can identify many forms of toxic language, including but not limited to threats, obscenity, insult, and identity-based hate speech, and explore appropriate machine learning methods to train a binary classifier for each of the following criteria:

1. toxicity

2. severe toxicity

3. obscenity

4. threat

5. insult

6. identity-hate

Implement multiple machine learning algorithms to determine which machine learning algorithm is best suited for a specific application. Perform data pre-processing and feature engineering to get the greatest accuracy for these algorithms.

Furthermore, I have made an additional effort to answer to the following questions as part of this project.

- Finding the classifier with the best performance.
- Determine if the performance of the model improves if training and testing are done with datasets that are distinct from one another classifiers.
- Explore the specifics of what a classifier can tell us about a comment's text depending on the provided datasets.

1.3 The scope of the project

The primary goal of this project is to develop machine learning classification models to determine the level of toxicity in the given comments data; however, a more extensive study related to the enhancement of existing industry practises may also be included. In addition, the project seeks solutions with pre-processing and feature engineering strategies that would benefit a classifier and enable it to deliver higher accuracy.

The project's pre-processing, feature engineering, and classification approaches are described in greater depth in the background study.

Extensive information about the data, including its many characteristics, may be gleaned by exploratory data analysis (EDA), which in turn supports the selection of appropriate pre-processing and feature engineering methods. To simplify EDA for the end user, a data science methods like data visualisation will be employed. The data may be better understood and utilised in the project's development with the help of the many visualisations offered by this project, such as pie chart, bar graphs, distribution plot, and word clouds.

The project not only assesses the outcomes, but also gives the submission file, which includes submission ids from the testing dataset and a classification of the comment's toxicity from the model. Each type of classification method has its own different submission file.

In the end, the project provides a comprehensive report, the program's source code, and any necessary documentation alongside with additional effort of interactive web-based application. This project will be hosted not as a website, but as an interactive Jupyter Notebook. There are many other classifiers that can be used with this dataset, but this project will focus on finding the optimum machine learning method for the task at hand.

The main constraint of this undertaking is time. With given deadline to complete a final report, it is vital to implementation of the code and other supporting documents to keep the project on schedule. Learning new skills in order to use these libraries and packages in my application will take time.

1.4 Approach and Assumption

I used the agile technique to plan my approach to this project. Since I tackled the project in short, focused bursts known as "sprints," using iterative development cycles to complete its many components, this methodology was ideal. At the end of each sprint, I was able to assess my progress, speak with my supervisor if necessary, and then plot out the next sprint based on the time remaining to finish the project and add the important functions. This approach made it easy for me to adapt updates and new features I found while discovering different Python libraries.

Having time as the key limitation also allowed me to adjust my development process as required. Due to diagnosed with illness, I wasn't able to put in as much time on my project as I had hoped each week. However, I was able to mitigate this deduction by removing higher-risk items in the following sprint thanks to the agile methodology I was utilising. Also, I didn't think it would be necessary to do much work cleaning the comments or performing any data pre-processing or multiple feature engineering. This was not the case, however, as some of the data provided in the comment was of low quality and not all feature engineering techniques were compatible with all classification models. Since I wanted to train a model with clean comments and improve accuracy by employing various feature engineering techniques, I had to devote extra time to cleaning the comments.

The classification models could have been built using R, SAS and other similar languages and tools but I specifically have use python because I have more knowledge and experience of python academically and industrial as well. Which eventually saved my time during the development rather than learning a new programming a language and tool. Using python (3.9.6), I was able to express myself in codding manner as well. One of the main benefits of using python was It has vast number of libraries and packages that are very powerful for task such as classification in machine learning.

Jupyter notebook IDE was crucial to the creation of this app. I used Anaconda to set up a working environment free of the project's dependencies, and I used Jupyter notebook to generate the required data visualisations, pre-processing, feature engineering, and classification. Due to the computational demands of the LSTM model, I utilised Google Colab to create a neural network for the classifier and then exported the model for future usage in my jupyter notebook. At last, I used gradio to turn my code into a simple web app that can tell you a toxicity of the comment.

1.5 Summary of Outcome

The finished version accomplishes the set goals and correctly categorises the comment's toxicity. A record of each classifier's efforts and how it has categorised the test dataset before submission. In addition, the web app provides a simple UI with helpful text boxes and buttons to enhance the user experience, and it allows users to supply a built model with any comment so that it can categorise the toxicity of the comment.

1.6 Structure of the dissertation

This chapter provides background information on the topics covered in the dissertation.

Chapter 1 – Introduction

Provides the brief information about the topic, aims and objective, approach

Chapter – 2 Background and Literature Review

Background study of the topic and methodologies used in the project.

Chapter 3 – Methodology and Implementation

This chapter contains details on the attempts carried out as part of this project. This chapter begins by detailing the data sets to which the models are applied. Furthermore, the implementation of model's details and parameters used may be found here

Chapter 4 – Evaluation and Results

This chapter deals with the results of implemented models, how well the model performed.

Chapter 5 – Conclusion and Future work

This chapter deals with the analysis of the project and what things can be added in the future.

Chapter 6 – Reflections

This chapter talks about what I have learned through out the project in terms of field of AI.

Chapter 7 – References

This chapter contains bibliography.

2 Background and Literature Review

For a complete comprehension of the issue, some historical context is required. This chapter provides an introduction to the principal factors upon which the rest of the work rests. This section contains mainly technical vocabulary, with adequate explanations, representing the cross-disciplinary nature of the topic under discussion.

As a preliminary stage, it defines the issue at hand and provides some insight into what classifies as toxicity and associated concepts. In addition, the chapter examines why toxicity detection is so challenging. Then the technical context is provided, which covers the fundamentals of neural networks and related technologies. There is a section on pre-processing that details the methods that were used to prepare the data for analysis. The chapter concludes with a discussion of the many assessment techniques that might be used.

2.1 Context of the project

The Toxic comment classification task, as defined by Kaggle, is "developing a model that classifies toxic remarks into six distinct categories such as threat, obscene, insult, toxic, identity hate, and severe toxic."

Jigsaw and Google (both of which are part of Alphabet) have formed a research group called the Conversation AI team to create software to enhance communication in virtual spaces. Negative online behaviours, such as poisonous remarks, are a subject of study (i.e. comments that are rude, disrespectful or otherwise likely to make someone leave a discussion). They have created several types of public Perspective API-served models so far, and toxicity is one of them. However, existing models continue to make error, and they don't provide consumers the option to narrow their searches to only the categories of toxicity they care about (e.g. some platforms may be fine with profanity, but not with other types of toxic content). In this competition, a user has been asked to create a multi-headed model that outperforms Perspective's existing models in detecting toxicity, such as threats, obscenity, insults, and identity-based hate speech. A user will be working with a dataset consisting of user feedback on proposed changes to the Wikipedia talk page.

Considering that the original labelling was done via crowdsourcing, there is a chance that the ratings applied to the dataset by its various users are inaccurate. Kaggle uses the mean column-wise Area Under the Curve (AUC) for Receiver Operating Characteristics to rank the entries in this challenge (ROC). The evaluation is then calculated as the mean of the six projected columns.

2.2 The problem of toxicity on the internet

Toxic online behaviour is defined as something which does more harm to others' or one's own sense of self-worth than it does to one's own development as a person (Lapidot-Lefler & Barak, A, 2012). According to Duggan's research, 66 percent of U.S. citizens have personally experienced internet harassment (Duggan, 2017). According to a recent survey by Google, cyberbullying is the leading cause of worry for teachers regarding student safety in the classroom (Google Survey, 2019).

When people share their thoughts, feelings, and experiences online, they run the risk of being exposed to potentially toxic content in front of the whole world via the web. The problem of cyberbullying, trolls

and spammers that make offensive posts and comments online, and it's become easier than ever to find inappropriate material on social media platforms Because of the fact that most people conceal their identity on social media by using a false username and photo. That said, there have been there have been several instances of people deleting their social media accounts or otherwise avoiding online communities due of toxic behaviour. Turn off the ability for others to comment on your posts and photographs. As an additional note, there have been severe occurrences that have led to the loss of innocent life (Internet Troll, 2019).

Cyberbullying and online harassment often lead to low confidence, low self-esteem, and heightened feelings of worthlessness in those who are targeted. Suicidal ideation, along with other unpleasant feelings including melancholy, frustration, wrath, and even fear (Smith, Peter K, et al., 2008).

However, the most difficult aspect is the reality that only a small percentage of people experience toxicity from specific types of content. Bullying may not be immediately apparent to an outsider if the incident being used as the basis for the bullying is only known by a small group of people. which means that the comment itself isn't the primary factor in making a verdict; the judge's background matters as well.

Making a machine learning / deep learning model to automatically categorise toxicity is also a challenging task. To this day, such an algorithm would continue to be trained on data labelled by humans, so inheriting their inherent bias. However, there are a few more issues with machine learning algorithms besides reliance on subjective data (J. Angwin & H. Grassegger, 2017).

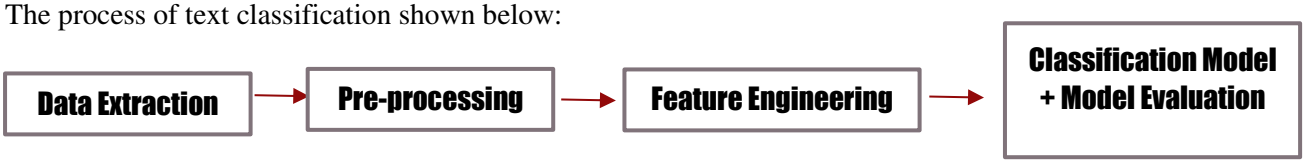
The main topic of discussion in this section is the harmful effects of Toxicity. However, this type of content is insufficient when thinking about social media. Other than text, toxicity can manifest in a variety of ways. This includes things like pictures, videos, and audio files. In extreme circumstances, toxicity can take on a variety of various forms, some of which are not readily apparent to the naked eye. A combined solution that can assess all of these elements is required to filter out different types of toxicity.

2.3 Natural Language Processing (NLP) and Text Classification

Natural Language Processing (NLP) is a subfield of AI that dealing with processing human-readable language and helping computers comprehend it. The primary purpose of Natural Language processing is to extract, analyse, and interpret human languages in order to make sense of them. The majority of NLP approaches depend on ML algorithms to analyse data and extract meaningful insights. NLP's job is to identify natural linguistic forms from provided material and transform them to computer intelligible languages. This is accomplished by using multiple algorithms to determine the meaning of each phrase in a text and gather relevant data from it. Human languages have words with emotional connotations. When employed in various settings, the same statement might have distinct meanings. As a result, computers must grasp and evaluate the significance of a specific statement in light of its context. Text processing is accomplished using several NLP stages. Sentiment analysis, personal assistant apps such as Siri, Alexa, Google, and others, language translators, and so on are examples of NLP applications (Krishna Dubey, et al., 2020).

The Text Classification process is divided into many subphases, each with its own relevance and requirement. Text Classifier is made up of six fundamental sub-components: Data gathering, pre-processing, feature extraction, feature selection Classifier construction, performance assessment (Sebastiani, 2002).

The process of text classification shown below:



2.4 Supervised learning And Text classification

The field of Machine Learning focuses on the development of algorithms that learn from data examples and then use that knowledge to the prediction or selection of new data. "**A computer programme is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, increases with experience E**," according to Tom M. Mitchell (Mitchell, 1997).

It entails the creation of computer programmes that adapt or learn when exposed to fresh data, similar to data mining. Both systems sift through data in search of patterns. Data mining, on the other hand, collects data for human comprehension, whereas machine learning detects patterns in data and adjusts programme operations accordingly. Machine learning is always based on findings or data, actual experience, or instruction. So, in overall, machine learning is more about learning to perform better in the future based on prior experiences (Subbu, 2017).

When this algorithm learns using input data, also known as training data, with known target answers or labels, which might be a continuous or categorical data, this is referred to as supervised learning.

A model is created through training or learning and predicts the proper answer when given a fresh example. The supervised method is further subdivided into two components: classification and regression. The method in classification predicts the class into which the provided test data fall, while the technique in regression forecasts a quantitative value for the target variable. For example, we may consider investing as a classification issue (will the stock rise or fall) or a regression challenge (how much the stock rise) wherein we would like the machine to learn explicitly how to make investment decisions in order to maximise wealth (Subbu, 2017).

Many data scientists are researching data classification, and the need for a data classification set is expected to increase in the future for a variety of reasons, one of them is to recognise antisocial online conduct, antisocial community members, or those who behave oddly or look hazardous (Cheng et al., 2014). Classification enables the research of worldwide available on social networks in order to collect special knowledge obtained from hundreds of millions of users worldwide; thirdly, for analysing media produced in social communities, such as images, videos, sound, and text, and grouping users in connection to their areas, connections of friends, hobbies, activities, and professions. When a training set of examples with class labels is provided, the fundamental purpose of text classification is to recognise and assign the specified class to a chosen instance. Classification algorithms are machine learning data-processing characteristics that enable multi-class text-classification (Alpaydin, 2010).

An automated method of classifying texts according to their content is called text classification. Due to the proliferation of online resources, there is a pressing need for categorization schemes that can facilitate the efficient retrieval of relevant data. Typically, the system is made up of a feature selector and a text classifier. A document is mirrored in a feature selector, which is often a feature vector. The feature vector is sorted into the appropriate category by the classifier. Many methods have been discovered via research to be useful in feature selection for both increasing accuracy and decreasing the

dimensionality of the feature vector, which in turn decreases the amount of time needed to compute results (Liao S. & Jiang M, 2005).

Text classification techniques are highly prevalent nowadays. There is now a large lot of study being undertaken in the field of text classification. Text classification using machine learning may be accomplished using several machine learning algorithms that have been developed for these tasks over the years and have proved to produce high prediction accuracies and are therefore extremely dependable. Some of the most prominent algorithms for this purpose include Naive Bayes Classification methods, Support Vector Machines, and Deep Learning algorithms that employ architectures such CNN and RNN (Gurinder Singh, et al., 2019).

Classification algorithms are distinct data-processing elements of machine learning. When a training set of objects with class labels is provided, the goal of classification is to recognise and assign a predetermined class to a chosen item. Machine learning methods are now backed by huge data processing frameworks, which often comprise specialised libraries (Alpaydin, 2010). Latent Semantic Indexing (LSI) is one method that uses idea frequency rather than the frequency of individual words. Despite using these methods, the feature matrix remains complex, necessitating a powerful classifier (Xu J & Wang Z, 2004).

2.4.1 Logistic Regression

The supervised classification approach includes logistic regression. This algorithm has grown in popularity in recent years, and its application has expanded significantly. This method is used to classify class depending on logistic function (Liu YY, et al., 2011).

Logistic Regression is a popular tool in the fields of classification and predictive analytics. By analysing a given collection of independent variables, logistic regression may predict the likelihood that a given instance belongs to a specific class. Due to the nature of the result being a probability, the dependent variable is limited to values between 0 and 1. To find a relationship between two variables (x and y), logistic regression makes use of the mathematical logistic function. Logistic Regression models, like Linear Regression models, produce an input feature weighted sum (including bias term). On the other hand, this Regression model uses the logistic function to provide a derived value as its output rather than a simple numerical value (Preethi, 2018).

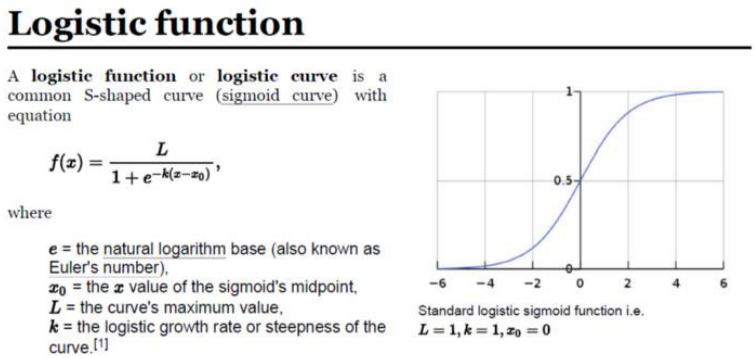


Figure 1, Logistic Function (Abhigyan, 2020)

2.4.2 Neural Network

As a machine learning approach, deep learning trains computers to mimic the way that people learn best: by observing and emulating examples. Many deep learning techniques rely on neural network topologies, thus the alternative name "deep neural networks" for these types of models. The amount of hidden layers in a neural network is often used as a proxy for how "deep" a network is thought to be. Deep networks may have as many as 150 hidden layers, whereas traditional neural networks have only 2. By using vast amounts of labelled data and neural network designs that can acquire attributes directly from the data, deep learning models can be trained with little to no human intervention in the feature extraction process (Mathworks, 2019).

A mathematical model based on the structure of human brain networks is called an artificial neural network. The neurons and connections among them make up this structure. Typically, there are numerous layers to their architecture. Numerous natural language processing (NLP) tasks have been successfully completed using deep learning based neural network models. These include parsing (Richard Socher, et al., 2013)statistical machine translation (Jacob, et al., 2014),sentiment categorization and many more. Learning a distributed representation of sentences using neural network models involves little training on an external domain and achieves good results in applications such as sentimental analysis and text classification (Chunting Zhou, et al., 2015).

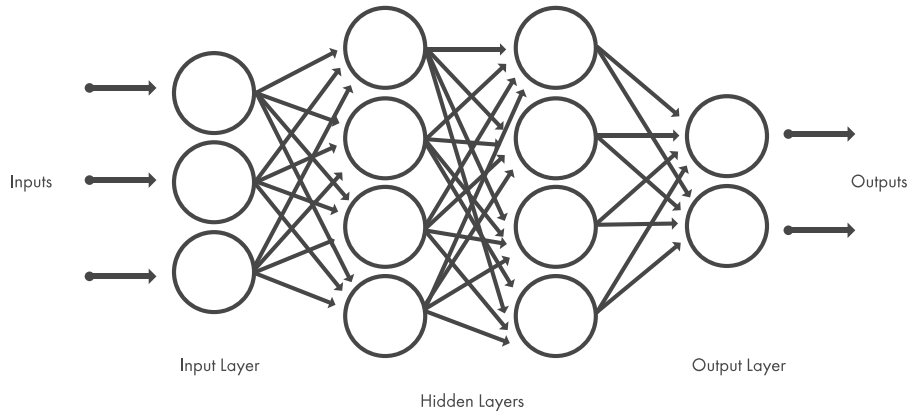


Figure 2, This is a basic example of a neural network's structure. It has three layers: input, output, and a concealed layer in the middle, (Mathworks, 2019).

-**Input layer** – This is the layer where information is introduced to the system.

- **hidden layer** – This layer has activation functions; the layer processes the output of the preceding layer (activated). The result of the input layer, in this example. Multiple hidden layers are possible in a neural network.

- **output layer** - where the results of hidden layers are used to calculate outputs of the desired label.

A basic example of a neural network's structure can be seen in above Figure 2. The current model uses an input layer of 3 neurons, a hidden layer of 5 neurons, and an output layer of 2 neurons. Data is shown flowing from the input layer to the output layer, with no connections occurring inside a layer or counter to the ow direction. One term for this kind of network structure is "feed-forward neural network" [55]. All the neurons in one layer are interconnected with the neurons in the next layer in this example. Full connectivity means that such levels are referred to be fully connected layers.

2.4.2.1 Long Short-Term Memory (LSTM) Neural Network

Recurrent Neural Network (RNN) is a type of Neural Network in which the output of the previous step is supplied into the current stage as input. This is a temporary memory used to process Sequential data. Long short-term memory (LSTM) networks are a subclass of RNNs. Basic RNNs have limited ability to learn dependencies over extended time periods. Most RNNs are trained via backpropagation, which may cause issues with the disappearing or exploding gradient. Due to the aforementioned issues, network weights tend to become either very tiny or extremely big, reducing the network's usefulness in applications that call for it to learn long-term associations (MathWorks, 2019).

Attractive about the LSTM model is the fact that each cell state may modify its own state by adding or removing gates. This is practical since it enables the model to retain the comment's linguistic context and draw conclusions from words throughout. For each of the 100 words in the remark, the LSTM model used a single densely linked layer with 50 units and 24 hidden states (Colah, 2018).

A bidirectional LSTM, often known as a biLSTM, is a sequential processing model that includes two LSTMs, one accepting input in one way and the other in the other. BiLSTMs effectively improve the network's accessible information, boosting the context for the method (e.g. knowing what words immediately follow and precede a word in a sentence).

It enhances the quantity of input available to a neural network. As the name implies, the model not only takes in information from previous states, but it also analyses data from both past and future states, which improves the neural network's capacity to grasp the context of the input. A bidirectional LSTM layer, in comparison to a regular LSTM layer, adds another set of LSTM cells to processing inputs in reverse order. Two sets of cell outputs are concatenated and sent to the subsequent layer (Mike Schuste & Kuldip K. Paliwa, 1997).

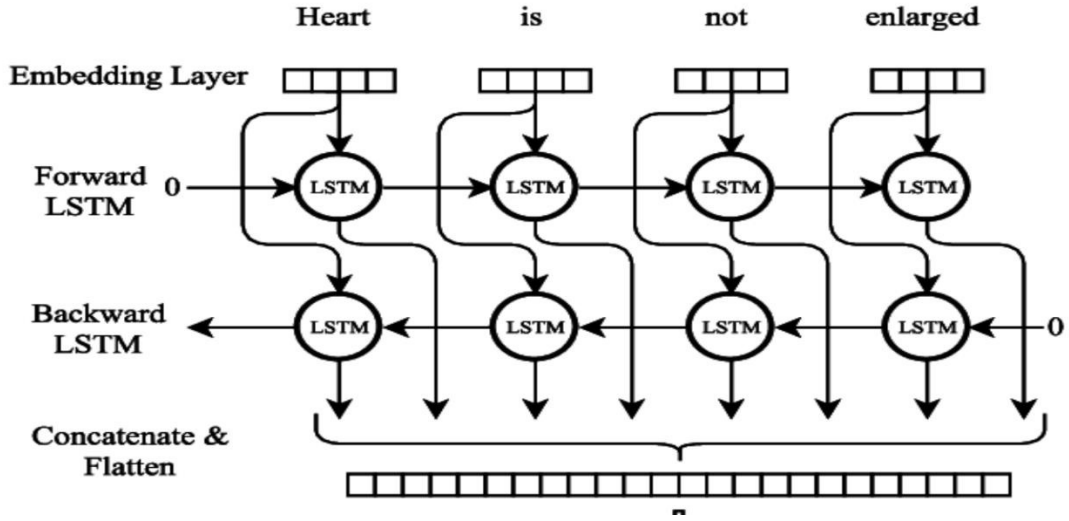


Figure 3, Bidirectional Long Short-Term Memory Networks (Cornegruta, 2016)

2.5 Literature Review

Textual aggression is a complicated topic that is being studied and addressed by several knowledge disciplines. This literature review of on a computer science viewpoint on textual toxic classification, a newly developing field. The scientific research of automated recognition of toxic writing using information technology approaches is currently expanding. Several pieces of linked literature are employed in this research to convey various sorts of toxicity.

Most social media sites employ some type of content moderation in an effort to eliminate the spread of harmful materials. There has to be fair, scalable tools in place to identify potentially harmful information while it is being posted online. Users will have more faith in these algorithms if they can see that the harmful content classification is tied to a specific section of text. It's important for individuals to have access to social media sites that are devoid of harmful content in order to foster constructive dialogue.

Automatically detecting textual toxicity is a well-studied problem. Every day, a flood of comments and tweets are published online, making it necessary to monitor and report any potentially offensive content and moderators can no longer handle abusive comments effectively by themselves. Automation is very useful in these kinds of situations since toxic and inappropriate comments can be reported to moderators so they can be removed.

Internet hostility and trolling behaviour identification are two issues that have attracted the attention of many academics. Before, scientists relied mostly on lexicon-based approaches. A dataset of hate speech was gathered by Davidson, Warmesley, Macy and Weber (Davidson, T, et al., 2017) to better understand the issue. Authors made extensive use of user submissions. At first, they amassed tweets containing hate speech terms using a crowd-sourced hate speech vocabulary. After that, they turned to the crowd once more to classify some of these tweets as either including hate speech, containing merely foul language, or both. In the end, research proved that hate speech can be consistently distinguished from other forms of abusive language. They discover that tweets that promote racism or homophobia are more likely to be labelled as hate speech, whereas tweets that promote sexism are more likely to be labelled as offensive. Malmasi and Zampiere dug deeper into this data set (Malmasi, S & Zampieri, M, 2017). They discovered that failing to differentiate between profanity and hate speech was the greatest barrier to effective hate speech identification.

The private sector is not immune to this issue, however. Kaggle, a machine learning competition platform, has hosted multiple challenges aimed at identifying and analysing aggressive and poisonous speech on social media. The top-performing models on Kaggle included a wide range of methods for boosting model performance. These methods included back-translation and other pseudo-labelling approaches, as well as new language modelling subtasks.

Several previous research have utilised neural network methods to examine hate speech; Badjatiya et al., conducted extensive trials with different deep learning architectures for learning semantic word embedding, with the aim of identifying harmful remarks (Pinkesh Badjatiya, et al., 2017). A separate investigation utilised a YouTube video sentiment analysis technique. As a result of the abovementioned discussion, a deep neural network was suggested which resulted in a 70-80% accuracy rate (Alexandre Ashade Lassance Cunha, et al., 2019). Moreover, many neural network approaches are commonly used for contemporary literature has made substantial use of remark categorisation (Piotr Bojanowski, et al., 2017). Moreover, numerous neural network techniques are often employed for modern literature has made considerable use of comment classification. In addition, a large body of research indicates that supervised learning approaches are the most popular approach to cyberbullying detection. The effectiveness of alternative non-supervised methodologies and procedures for cyber-bullying identification has, however, been established (Sweta Karlekar & Mohit Bansal, 2018).

After delving into the many pieces of writing for this problem, it became clear that the early works had been the subject of various research. For instance, in 2018 Revati Sharma and Meet Kumar Patel used Neural networks, including CNN and RNN, through the application of word embedding techniques and a direct comparison of these networks, outcomes with complex CNN and main level neural network methods RNN. But In contrast to CNNs with word-level embeddings, LSTMs perform better in terms of accuracy and speed when trained with the same number of epochs (Revati Sharma & Meetkumar Patel, 2018). In another approach by Mujahed A. Saif in 2018, using logistic regression and RNN, LSTM among these 2 LSTM layers and 4 conv layers, has got a score of 0.9645 showing best accuracy (Mujahed A. Saif, et al., 2018).

Using Naïve Bayes Support Vector Machine, CNN, LSTM using TF-IDF as extraction feature to classify toxic comments, The best accuracy is achieved by utilising the NB-SVM model, which has an accuracy of 98.13% (A. A. Sagar & J. S. Kiran, 2008). Another study used multilabel classification techniques such as Nave Bayes, SVM, and KNN, as well as feature selection approaches such as Mutual Information, Odd ratio, and Chi Square to accomplish multilabel classification. The best F1score achieved by utilising the SVM approach using Chi Square as the feature extraction is 80.03% (A. Y. Taha & S. Tiun, 2016).

2.6 Libraries

The researcher widely uses Python, making Python one of the most commonly used programming languages. Developers may take use of the language's structure and object-oriented approach to produce code that is both readable and easy to follow, regardless of the project's size. Python libraries (packages) effectively simplify several crucial procedures, including data analysis and visualisation, information retrieval from the web, image processing, model construction for machine learning, and textual information (Pandey, K & Panchal, R, 2019). The main libraries that have been utilised for this project's implementation across all approaches are listed below.

1) Pandas
   Pandas is a Python library for data analysis and manipulation that is quick, powerful, flexible, and user-friendly. Data cleansing, manipulation, and transformation are made easy using Pandas, which finds widespread application in the context of data analysis. Pandas simplify the management of massive data sets. The best aspects of Pandas may be broken are Quickly explore and analyse data, Read many book formats Data cleaning and manipulation (Gholizadeh, 2022).

2) Numpy
   The NumPy library provides multidimensional arrays and is freely available to the public. A NumPy ndarray is a "n" dimensional array object that stores data consistently across all dimensions (Oliphant, 2006). NumPy is widely used in industry for array computations; for instance, a colour image's data may be stored in a three-dimensional matrix with a thousand-pixel resolution. Pixels are the building blocks of pictures, and we may change them to suit our needs. In this case, NumPy really shines. Even more sophisticated Python libraries rely on NumPy, like Pandas and etc. Compared to Python's List, NumPy is more effective in two key areas: speed and memory use. All sorts of mathematical operations, linear algebra, random number generation, and so on are available right out of the box (Fontaine, 2018).

3) Matplotlib
   In terms of data visualisation, Matplotlib is by far the most widely used library. Line graphs, scatter plots, histograms, bar graphs, and pie charts are just few of the simple types of charts that may be made with it. Every other visual library builds on top of Matplotlib. The Python programming language and its

NumPy numerical extension now include a charting module. It helps decision-makers spot trends, correlations, and patterns that might otherwise be hidden in textual data (Barrett, P, et al., 2005).

4) Seaborn

As a plotting library, Matplotlib serves as the foundation for Seaborn. Used to create engaging and understandable graphical representations of statistical data. Seaborn is an extension of the Pandas DataFrame with a focus on supporting category variables for displaying data. Seaborn provides facilities for picking colour schemes which help to uncover latent data patterns (Waskom, 2021).

5) Plotly

Using the PX methods in the plotly.express module, you may quickly generate whole figures. The best place to begin for making most figures is using Plotly Express, which is included within the graph library. The plotly.graph objects.Figure instance is returned by every method in Plotly Express since graph objects are used internally. Dash is another tool that may help you visualise your data, similar to Plotly. Plotly creates Dash and offers a framework for developing and delivering enterprise-grade Dash applications [17]. To make interactive dashboards using Plotly visualisations, you may use Dash (Batalla, J. M, et al., 2015)

The comparison between these three-visualisation library can be seen below in a table, since plotly is more advance and beautiful for the output, I have mainly used plotly for visualisation purpose in the project.

Features	**Matplotlib**	**Seaborn**	**Plotly**
Functionality	Mainly used for basic plotting using bar plot, scatter plot and etc.	Specialise in statistical data visualisation.	Used to create beautiful and interactive plots.
Visualisation	Well integrated with numpy and pandas in python.	More integrated working with pandas dataframe.	High level API for data visualisation.
Flexibility	Highly customisable and powerful.	Provides default themes which are mostly used.	Provide simple syntax for complex visualisations.
Animation	Not available	Not available	Richly interactive plots including animation in a single function call.

Table 1: comparison between Matplotlib vs Seaborn vs Plotly (Gholizadeh, 2022).

6) Scipy

Mathematical, scientific, and technological issues can be addressed with the help of the open-source SciPy package for Python. This opens the door to a broad variety of data manipulation and visualisation techniques implemented using Python. SciPy was built on top of the NumPy module for Python. In addition to optimization and integration and interpolation and eigenvalue issues and algebraic and differential equations and statistics, it offers algorithms for these and many more types of problems (Gholizadeh, 2022).

7) Word cloud

A "word cloud" displays a set of words in varying font sizes. When a word's size and weight increase, it means it's been given additional emphasis in the text. The advantages of this particular wordcloud library

over other are utilisation of arbitrary masks as a capability, being readily customisable due to using a stupidly basic technique (with an efficient implementation) and Pythonic in nature (Mueller, 2020).

8) Sklearn / Scikit – learn

It's a machine learning library for the Python. Following data preparation and manipulation using Panda or NumPy, machine learning models may be constructed using Sklearn / Scikit-learn, which provides access to hundreds of tools for modelling and pre-predictive analysis (Fontaine, 2018). It may be utilised to construct a wide variety of machine learning models, including those based on supervised and unsupervised learning, model accuracy cross-validation, and feature significance. Support vector machines, random forests, gradient boosting, 3 k-means, and DBSCAN are just a few examples of the classification and regression methods included, and the package is built to work with the NumPy and SciPy Python numerical and scientific libraries (Pedregosa, F, et al., 2012).

9) Tensorflow / Keras

TensorFlow was created as a Python machine learning package that is freely available to the public. It's versatile and useful for many purposes, but it's main purpose is in training and inferring deep neural. The tensors it employs allow it to conduct several operations on the same input. Included with Tensorow is a tool called TensorBoard, which facilitates graph visualisation and model learning. Insight into the model's nodes is a big assistance when it comes to fixing faults and improving the model's performance. Graph Dashboard is a potent instrument for investigating the tensorow model and providing a snapshot of the model's architecture and design. The TensorFlow APIs are organised in a hierarchy, with higher-level APIs constructed from the lower-level APIs. To develop and study novel machine learning algorithms, researchers turn to low-level APIs. tf. keras is an implementation of the Keras open-source API that works with the TensorFlow deep learning framework (Silaparasetty, 2020).

In Figure below we see the TensorFlow toolkit:

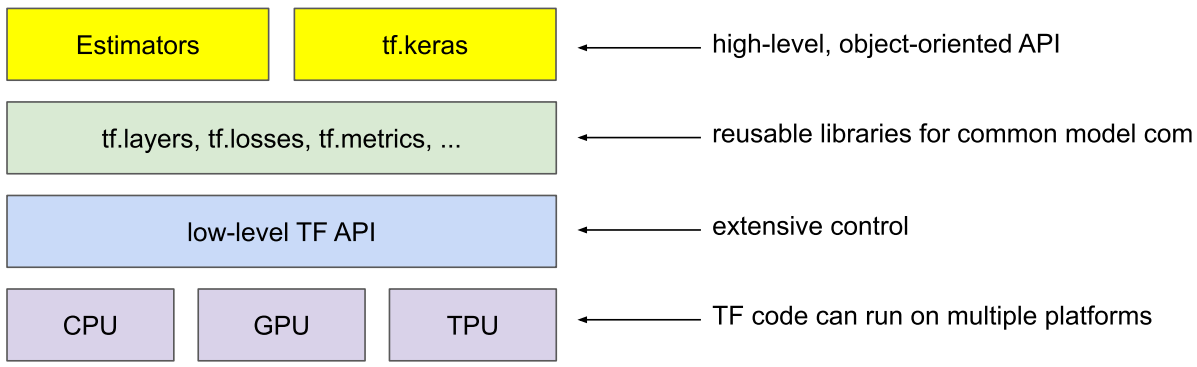


Figure 4 - (Google's Intro to TensorFlow, n.d.)

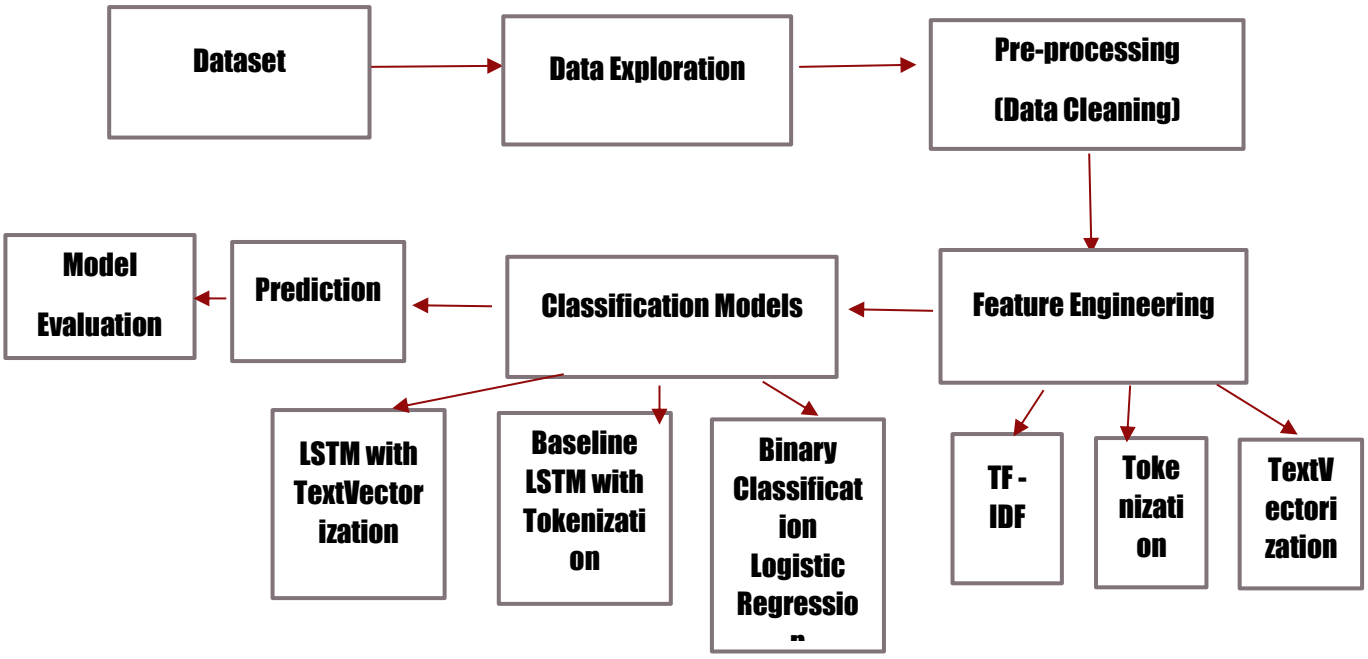
10) Keras

Keras is a Python-based deep learning API that is built atop Google's TensorFlow machine learning library. The potential for speedy experiments inspired its creation. Specifically, neural network models, which are the most common type of deep learning model, are the most common usage for Keras NLP and Computer Vision CV are just two of the many areas where Keras may be put to use (fcholle, 2020)

3 Methodology and Implementation

This section will outline the approach, as well as the overall methodologies and implementation of the code was built for this project.

The process of pipeline can be seen in the below



3.1 The dataset

Hundreds of thousands of Wikipedia comments have been provided in the datasets, all of which have been manually rated for their potential to be harmful to others. The various forms of toxicity are:

- Toxicity
- Severe toxicity
- Obscenity
- Threat
- Insult
- Identity-hate

train.csv - data used for training, including annotations and their respective binary labels.

test.csv - the test set, you'll need to estimate the likelihoods of toxicity for each of these remarks. In order to prevent manual labelling, the test set includes remarks that will not be factored into the final score.

sample_submission.csv - example of a well formatted submission file.

test_labels.csv - labels for the evaluation data; a score of -1 indicates that the item was not included in the final score calculation (Note: file added after Kaggle competition ended, but use of this will be helpful.)

The training.csv dataset contains 159,571 labelled samples of toxicity-rated Wikipedia comments. The dataset has seven columns: id, toxic, severe toxic, obscene, threat, insult, and identity hate, with all toxicity categories represented as Boolean values. The test.csv dataset has two columns: id and comment text, with comment text containing 153,164 comments to be labelled as toxic, severe toxic, obscene, threat, insult, or identity hate.

3.2 Data Exploration

As we can see in the figure below that training dataset has 8 columns having 159571 variables without any missing values.

Number of variables	8	Categorical	8
Number of observations	159571		
Missing cells	0		
Missing cells (%)	0.0%		
Duplicate rows	0		
Duplicate rows (%)	0.0%		
Total size in memory	9.7 MiB		
Average record size in memory	64.0 B		

Figure – 5, general information about training dataset, using pandas profiling library.

Meanwhile, comment text contains 157,865 different values, with a substantial association between toxicity and obscene, toxic and insult, severe toxic and obscene, identity hate and insult. More information regarding correlation will be given later in the data exploration section.

`id` has a high cardinality: 159571 distinct values	High cardinality
`comment_text` has a high cardinality: 157865 distinct values	High cardinality
`toxic` is highly correlated with `obscene` and 1 other fields	High correlation
`obscene` is highly correlated with `toxic` and 2 other fields	High correlation
`insult` is highly correlated with `toxic` and 3 other fields	High correlation
`severe_toxic` is highly correlated with `obscene` and 1 other fields	High correlation
`identity_hate` is highly correlated with `insult`	High correlation
`id` is uniformly distributed	Uniform
`comment_text` is uniformly distributed	Uniform
`id` has unique values	Unique

Figure -6 , overall information about columns.

Because the comment text column is our dependent variable, it would be fascinating to examine its insight. The maximum length of a comment is 5000 characters, whereas the median and mean lengths are 4570 and 368 respectively. Where 1563 different characters are divided into 8 distinct categories. Figures 7-10 below shows further information regarding the comment text overview insights.

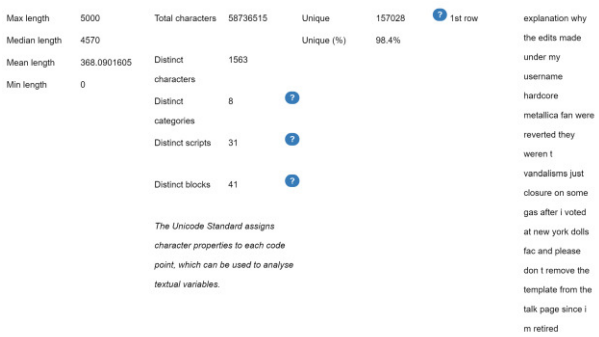


Figure -7 , overall insights of comment_texts

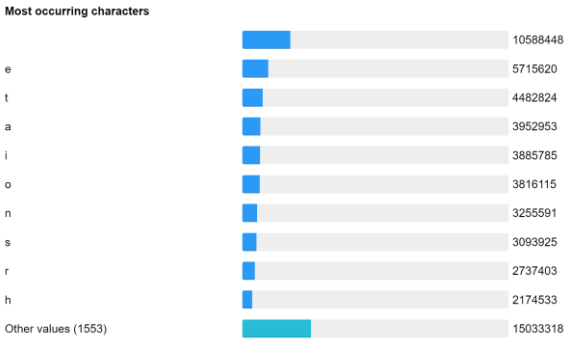


Figure -8 , most occurring characters

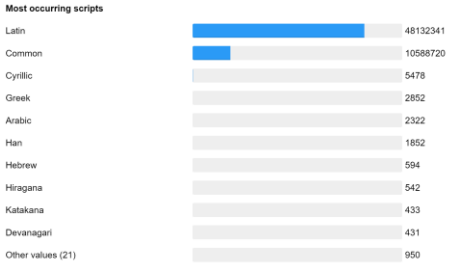


Figure - 9, Most occurring scripts

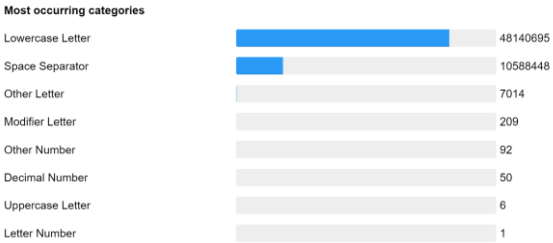


Figure - 10, most occurring categories

This data set is clearly unbalanced and slightly biased towards clean comments, which is to be expected given that toxic comments (where any label = 1) are often the exception rather than the norm. It will also be fascinating to examine the overall comments (toxic vs clean). All comments with a 0 label in any toxicity class are clean comments; all comments with a 1 label in any of the toxicity classes are toxic. Creating new variables where comment type_0 means all comments where particular comment type = 0 and comment type_1 means all comments where particular comment type_1 = 1.

(here comment type as 6 labels, e.g. toxic_0, toxic_1, severe_toxic_0, severe_toxic_1)

3.2.1 Exploratory Data Analysis (EDA)

3.2.1.1 Pie Chart:

A Pie Chart is a sort of graph that shows data in the form of a circular graph. The graph's parts are proportionate to the proportion of the total in each group. Each piece of the pie is proportional to the amount of that group in the overall group. The full "pie" represents one hundred percent of the total, whereas the pie "slices" represent segments of the whole.

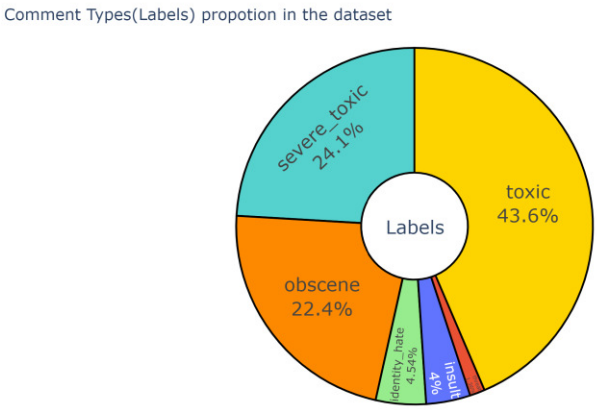


Figure – 11, Pie chart

As the pie graphic shows, the training dataset is significantly uneven, with the bulk of comments being toxic (43.6%), severely toxic (24.1%), and obscene (22.4%). In comparison to other comment types, the prevalence of threats, insults, and identity hate is low. This could be one of the reasons why our models are more biased toward toxic, severe toxic, and obscene substances. Because we have a smaller training dataset for other comment kinds. The training of models has been shown to be sensitive to class imbalance issues because the model might get overloaded by majority class instances, leading it to disregard minority classes [16].

Wikipedia is a freely modifiable collaborative platform, therefore discussions including toxic language are not representative of the bulk of the data in our collection. In a similar context, Ying Liu et al suggested that the high cost of manual labelling efforts might be to blame for this disparity. Neural networks are usually able to compensate for class imbalance with enough repetitions of the dataset (epochs). To mitigate the impact of a bad prediction on the model's performance, we may employ sampling procedures to shrink the size of the training set, or we can incorporate a penalty into the loss function.

After further analysis I have found that, dataset exhibits are class imbalance. The Training dataset has, 10.17% comments are toxic comments (any_label = 1).

For comments where toxic = 1,

There are 15294 toxic comments. (9.58% of all data.)

- From that, 1595 or 10.43% were also severe_toxic.

- From that, 7926 or 51.82% were also obscene.

- From that, 449 or 2.94% were also threat.

- From that, 7344 or 48.02% were also insult.

- From that, 1302 or 8.51% were also identity_hate.

- From that, 15294 or 100.00% were also any_label.

For comments where obscene = 1,

There are 8449 obscene comments. (5.29% of all data.)

- From that, 1517 or 17.95% were also severe_toxic.

- From that, 301 or 3.56% were also threat.

- From that, 6155 or 72.85% were also insult.

- From that, 1032 or 12.21% were also identity_hate.

- From that, 8449 or 100.00% were also any_label.

For Comments where insult = 1,

There are 7877 insult comments. (4.94% of all data.)

- From that, 1371 or 17.41% were also severe_toxic.

- From that, 307 or 3.90% were also threat.

- From that, 1160 or 14.73% were also identity_hate.

- From that, 7877 or 100.00% were also any_label.

For comment where any from the above label = 1,

There are 16225 any_label comments. (10.17% of all data.)

- From that, 1595 or 9.83% were also severe_toxic.

- From that, 478 or 2.95% were also threat.

- From that, 1405 or 8.66% were also identity_hate

The further analysis of comment overlapping shows that,

1 label: 39.2% comments have 1 label.

2 labels: 21.4% comments have 2 labels.

3 labels: 25.9% comments have 3 labels.

4 labels: 10.8% comments have 4 labels.

5 labels: 2.4% comments have 5 labels.

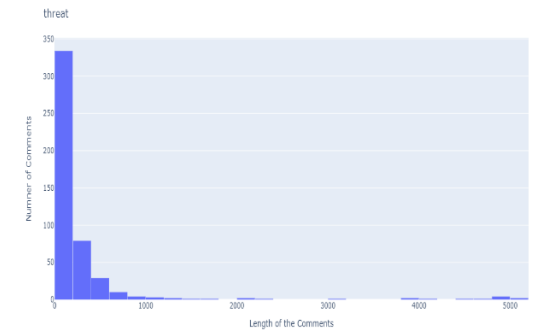
6 labels: 0.2% comments have 6 (all) labels.

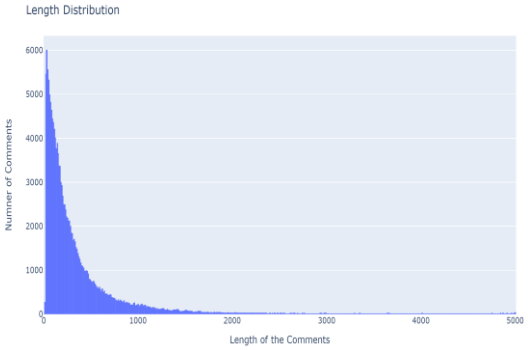
3.2.1.2 Distribution Plot:

Distribution plots evaluate the distribution of data sample graphically by matching the empirical data distribution to the theoretical values anticipated given a specific distribution.

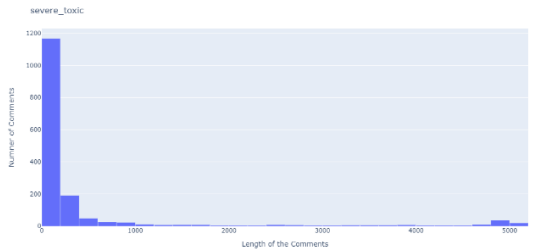
The histogram, shown in Fig. 4.5, is a graphical representation of the dataset based on the comment length distribution plot (from the figure below). The majority of the text lengths are under 500 characters, with some exceeding 5000. Below we can see the length distribution for each type of comments and most of them falls under 500 characters.

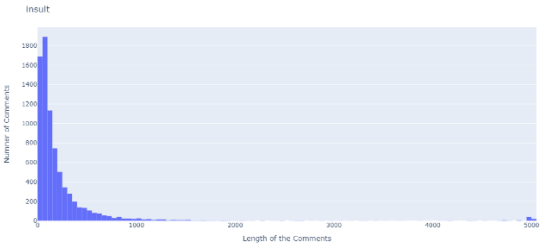
Figurers: 12-19

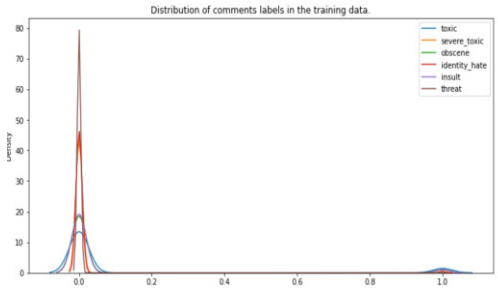


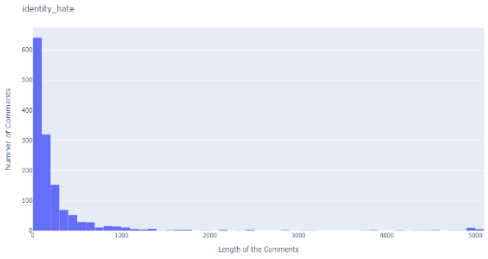


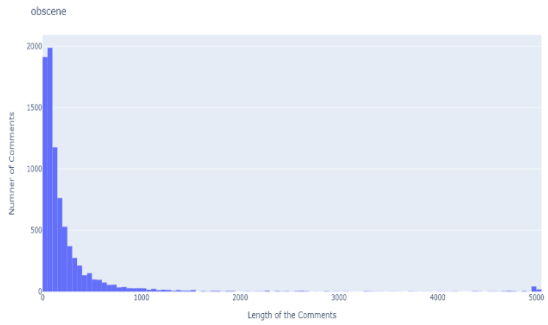
.

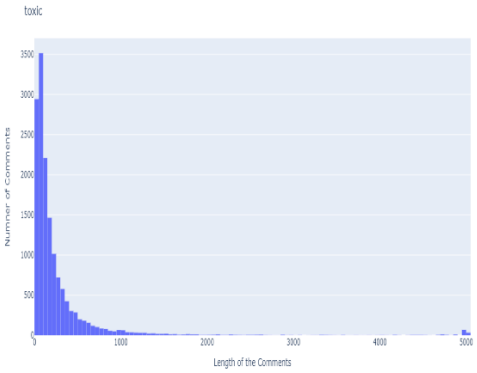












3.2.1.3 Correlation plot:

A correlation is a matrix that displays the coefficients of correlation between variables. Every cell in the table represents the relationship between two variables. A correlation matrix can be used to summarise data, as an input to a more advanced methods, or as a diagnostic for further studies.

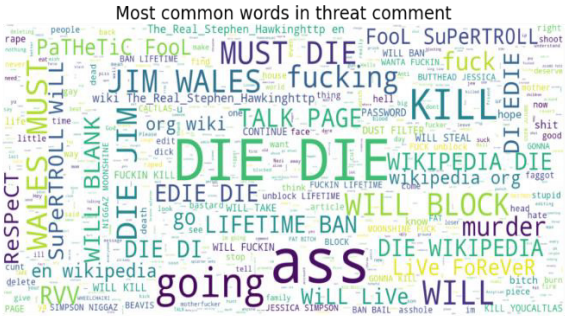
The correlation matrix of the categories is given in Figure - . The correlation matrix shows 'toxic' is correlated with ‘obscene’(0.68) and 'insult' (0.64). Threat has lowest correlation with all of the comment types. One of the reasons behind this could be unbalanced data and having low number of threat

comments. 'insult' and 'obscene' have a correlation factor of 0.74 This implies 'insult' and 'obscene' are highly correlated.

Figure –20 , Correlations

3.2.1.4 Word cloud

A word cloud is a basic yet useful interactive storytelling element for text processing that displays the most common term with larger and bolder characters and varied colours. The smaller the term, the less essential it is.



Figures: 22 – 26, wordcloud for each comment type.

3.3 Data Pre-processing

In most cases, the raw data set is not used when feeding data into a machine learning model. It may be required to reformat the data so that the model can make sense of it. In other circumstances, performance can be improved by the removal of extraneous features, the addition of new features, or the modification of existing functions. Data pre-processing refers to the collection of activities that occur before input of data into a model. This section provides a summary of the necessary steps taken to prepare the textual data that will be utilised in the rest of this project.

Pre-processing has the ability to improve the overall quality of a dataset and the quality of the dataset for Text Classification when applied to a given dataset. Performing the preparation phase allows for the dataset to be "cleaned" of noise, For Text Classification tasks, the quality of the dataset may be improved by the application of pre-processing techniques such as the removal of stop words and punctuation marks, as well as the performance of word stemming and lemmatization (Yaakov HaCohen-Kerner, et al., 2020).

3.3.1 Data Cleaning

Cleaning the data is a crucial step in every data science project since it improves the quality of the information being analysed. The value of our data is proportional to the insights we are able to draw from it (Guven, 2020).

Data cleaning, also known as data cleansing or data scrubbing, is the process of examining data for and fixing mistakes and discrepancies. Single data collections, such as files and databases, might have issues with data quality for a variety of reasons, including but not limited to typographical errors, missing information, and incorrect data. The need of data cleansing grows dramatically when combining data from several sources, as is the case in data warehouses, federated database systems, and worldwide web-based information systems. The reason for this is that there is often duplication of information among sources, although in different forms. Consolidating various data representations and getting rid of redundant information becomes required to enable access to correct and consistent data (Erhard Rahm & Hong Hai Do, 2017).

There are a number of criteria that a data cleansing method should meet. In the first place, it has to be able to spot and fix key problems when working with single data sources and when combining several sources. The method should be easily extended to encompass other sources with minimal effort and be backed by tools that reduce the need for human examination and programming. In addition, data cleansing should not be done in isolation from other schema-related data transformations that are informed by thorough metadata. It's important that mapping functions used for data cleaning and other data transformations may be stated in a declarative manner and reused across different data sources and query processing. In order to conduct all data transformation procedures for many sources and big data sets in a reliable and fast manner, a workflow architecture should be enabled, and this is especially important for data warehouses (Erhard Rahm & Hong Hai Do, 2017).

3.3.1.1 Data Cleaning using Regex

Regular expressions are sequences of letters that define a pattern of text that has to be located. These expressions are used in computer programming. They may be utilised for confirming test findings, processing text files in search of certain patterns, discovering keywords in emails or on websites, and so on. Regular expressions, often known as regexes, are an abstraction of keyword search that make it possible to recognise text based on a pattern rather than an exact string. Keyword search is the foundation of regular expressions. Common applications for regular expressions include scanning text files for a certain pattern using applications such as grep, vim, and Eclipse; verifying the content that users submit into online forms using Javascript; and processing text using general purpose programming languages. There are tens of thousands of bug reports because regular expressions are difficult to learn, maintain, and debug, despite the fact that they are extremely strong and diverse (E. Spishak, et al., 2012).

In order to clean up the data for this project, I made extensive use of regular expressions. Regex was used for this section primarily to lower the text, remove punctuation from the comments, remove URL from the comments, remove html tags from the comments, remove leaky elements such as IP addresses and usernames from the comments, and finally replace short words with their full-length equivalents.

To make the text data in same case, we will be lowering the case. With the help of Python's. lower() method, I've written a function that would transform comments into lowercase text, as seen in Figure 29 .

```
def lower_text(text):
    return text.lower()
```

(Figure -29)

To remove punctuations from the comments, I have used the following method that can be seen in the figure 30.

```
def remove_punctuation(text):
    regex = re.compile('[' + re.escape(string.punctuation) + '0-9\\r\\t\\n]')
    text = regex.sub(" ", text)
    return text
```

(Figure 30)

I used the following method (figure 31) to remove URL from the comments by constructing a function of it, deleting URL will eliminate noise of hyperlinks from the dataset.

```
def removing_url(text):
  text = re.sub(r"\b(?:(?:https|ftp|http|www)://)?\w[\w-]*(?:\.[\w-]+)+\S*", '', text, flags=re.MULTILINE)
  return text
```

(Figure 31)

Following that, I wrote a function that replaces the common words, removes html elements and IP addresses from comments; the code for this function is shown in Figure 32.

```
def remove_common_words(text):
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"can't", "cannot ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\'re", " are ", text)
    text = re.sub(r"\'d", " would ", text)
    text = re.sub(r"\'ll", " will ", text)
    text = re.sub(r"\'scuse", " excuse ", text)
    text = re.sub('\W', ' ', text)
    text = re.sub('\s+', ' ', text)
    text = text.strip(' ')
    return text
```

```
def remove_html_tags(text):
    html = re.compile(r'<.*?>')
    text = html.sub(r'',text)
    return text
```

```
# remove leaky elements like ip,user
def remove_ips(text):
    text=re.sub("\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}","",text)
    return text
```

(Figure 32)

Following that, I built a new function named clean comments in which I placed the previously developed methods, and I am now applying these methods to the comment dataset for data cleaning using the .apply() method of Python and the lambda function.

3.3.2 Feature Engineering

3.2.2.1 TF – IDF Technique

The goal of feature extraction is to minimise dimensionality and eliminate unnecessary information in order to enhance the efficiency and effectiveness of classification algorithms. Pre-processing improves the quality of input data when using ML or other statistical approaches; hence, pre-processing seeks to turn free-text review phrases into a group of words while also increasing their semantic value. This procedure consists of three subtasks (Chaudhary, 2020).

The TF-IDF algorithm has several phases, where we load the data followed by tokenizing where in this phase every word in the document is split into tokens and then stored it as a Tlist.

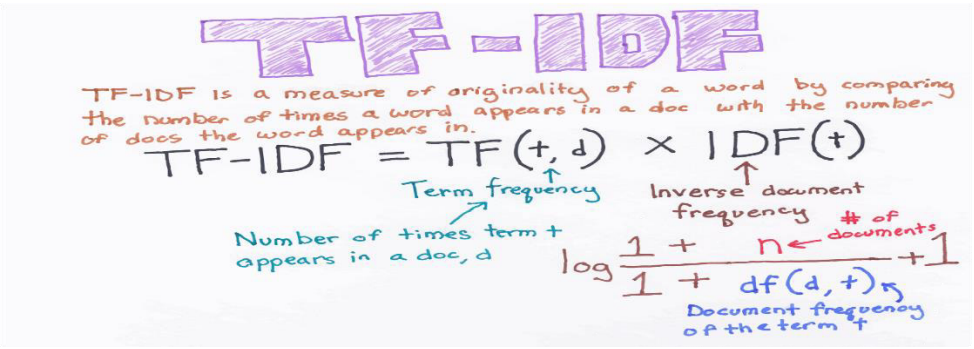


Figure - 33 , TF-IDF (Chaudhary, 2020).

3.2.2.2 Tokenization

The text is "tokenized" by the Tokenizer, as the name implies. The term "tokenization" refers to the method by which a string of text is broken down into its constituent parts (character, word, sentence, etc). The remark is split up into rows, with each word serving as input to the neural network. Tokenization is the process through which the remark is broken down into its constituent parts. On this job, you'll need a set of guidelines for how to divide the text and what to maintain as tokens. However, tokenization does not follow any established norms. Implementation decisions are extremely context and data dependent. A minimal tokenizer would just delimit tokens at whitespace. The comma and the preceding word would become one token in this case, which is problematic. Tokenization through keras.utils.pad sequence has been employed in the feature engineering of a baseline model. Pad sequence adjusts the duration of each sequence to be uniform (length of the longest sequence or provided override) (raghakot, 2015).

We then load the vocabulary from the train comments into the word tokenizer and use the pad sequence function to ensure that all of the sequences are the same length. I've tokenized both the training and testing sets because we'll be using test data for the prediction phase.

This is a crucial process for introducing new texts into the structures of this work's networks. Tokenization separates out each word into its own "token." Separate tokens are created for each unique character. However, emoticons are an exception since their entirety is treated as a single token.

3.2.2.3 Vectorization

A pre-processing layer which maps text features to integer sequences.

In order to control text in a Keras model, this layer provides several fundamental controls. Token indices (1D tensor of integer token indices) and dense representations (1D tensor of float values encoding data about the example's tokens) are generated from a set of strings (1 string Equals 1 example). Inputs in natural language are intended for this layer.

The layer's vocabulary must be provided during build time or acquired through adapt (). Following adaptation, this layer examines the dataset, counts how often each string value appears, and generates a vocabulary. Depending on the settings for this layer, the vocabulary size is either unbounded or restricted, and when the number of distinct input values exceeds the limit, the most common phrases are used to fill in the gaps.

I'm adjusting the vocab value of the trained comments so that they may be vectorized in TextVectorizer.

Next, I am shuffling the data in batches of 24 using textvectorizer to prepare it for the Bidirectional LSTM model, and then I will divide the dataset into training and validation sets using an 80%/20% split. To demonstrate how TextVectorization works, have a look at the sample we've provided below:

Comment: ‘explanation why the edits made under my username hardcore metallica fan were reverted they weren t vandalisms just closure on some gas after i voted at new york dolls fac and please don t remove the template from the talk page since i m retired’

Vectorized:

<tf.Tensor: shape=(45,), dtype=int64, numpy= array([651, 78, 2, 129, 134, 178, 32, 637, 4249, 10741, 980, 88, 323, 50, 2116, 23, 10844, 55, 6507, 16, 63, 2612, 151, 4, 2778, 38, 122, 1137, 15517, 2643, 6, 48, 61, 23, 240, 2, 354, 34, 2, 40, 30, 147, 4, 71, 3226])>

3.3 Model Implementation

In this chapter, we will go into deeper depth about the structure of our categorization model and the methods utilised for individual models.

3.4.1 Logistic Regression

The goal being to categorise each remark type, I have concentrated on developing a binary multiheaded classification model with logistic regression. I'm using the TF-IDF feature engineering approach for logistic regression, where I'm first vectorizing the words and characters by adding them to a vocab, and then stacking them as a training feature for the model.

The chosen classification approach used 10-fold cross-validation to train and evaluate the data. - Finding the mean categorization success rate for the test set. Following is an example presentation of the formula for the average accuracy of multi-class classification: (Sokolova and Lapalme, 2009)

$Accuracy = \sum tpi+tni\ tpi+fni+fpi+tni\ l\ i=1\ l \times 100\%$

, where tpi are positive instances of the classification, fpi are negative examples, fni are false negative examples, and tni are true negative examples, and l is the number of classes.

Accuracy in classifying data is measured by the proportion of instances when the observed label matches the anticipated label as a percentage of the entire test data corpus.

Here, I'm employing solver ='sag' for the classification model and 'roc auc' for the evaluation metrics. I am training each comment type and making predictions for the test dataset using a for loop. To verify my results, I added this dataset to the Kaggle submission file.

3.4.2 LSTM Neural Network

The LSTM Recurrent Neural Network was employed for this task. Though RNN is designed to work with sequential data, training the model might be problematic due to the length of our comments. Since our project demands a neural network that should not suffer from the vanishing gradient problem, we decided against using a simple RNN, which has this issue among others. Since the learning rate "n" is already less than 1, the change in weight "w" tends to grow less and smaller when we perform Back-propagation and compute the shift in error with regard towards the weights (de/dw). Since the total weight change is so negligible, the newly achieved weight is almost identical to the old one. This means that the neurons within the first layers of a neural network take much longer to learn than those in the later levels. As a result, training time is longest for the network's first few layers. This causes the training process to take too long, resulting in a drop in the Model's Prediction Accuracy (Krishna Dubey, et al., 2020).

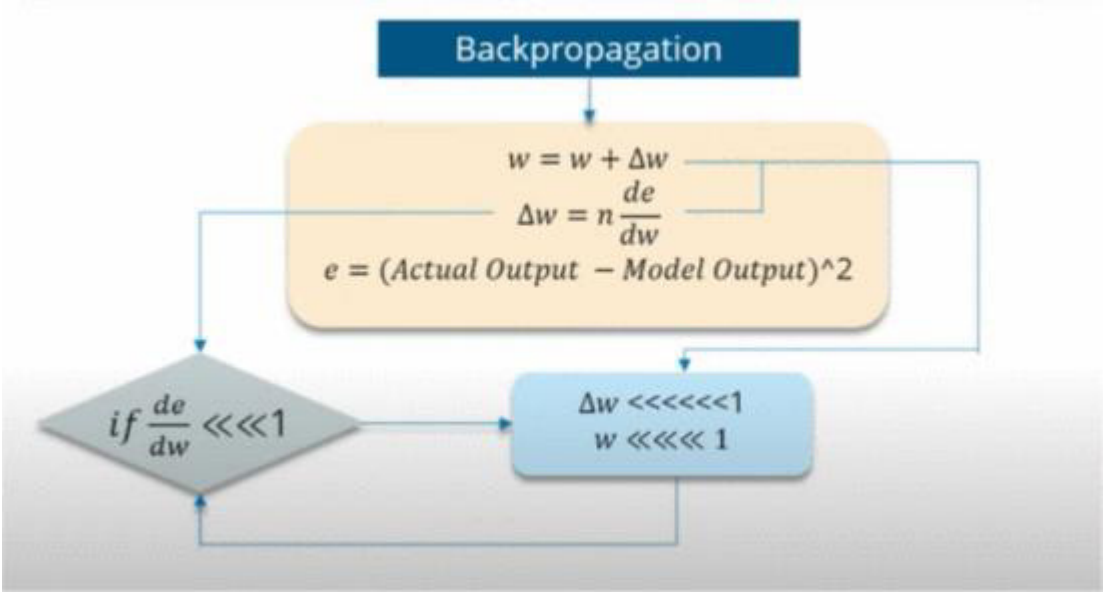


Figure -34, Vanishing gradient problem in Recurrent Neural Network (RNN) (Krishna Dubey, et al., 2020)

3.4.2.1 LSTM baseline with Tokenization

I next utilise the Keras framework, which is based on the Tensorflow backend and has Graphics Processing Unit (GPU) support, to build a simple recurrent neural network. For training a neural network, I have used Google colab high ram and GPU because the sheer volume of calculations might overwhelm the CPU. Experimental results show that the GPU accelerates our work by a factor of more than three when compared to a conventional four-core, eight-thread central processing unit. Keras, on the other hand, is a Python-based open-source neural network architecture that is capable of integrating with a wide variety of machine learning backends. It retains the power of a standard machine learning framework while being intuitive for newcomers and easy to modify. The RNN is based on its Sequential Model.

Our model's first layer is the input/embedding layer. Because of our 100000-word vocabulary, 100-word padded phrases, and 100-length embedding vectors for each word, this input/embedding layer has a dimension of [100000 * 100] for each input of size 100.

Our neural network's second layer is a bidirectional long short-term memory (LSTM) layer; I've programmed it with 32 cells; as a bidirectional layer, it will contain 64 nodes in total. In the following phase, I've planned for a 10% dropout rate and a 10% recurring dropout rate. In this layer, a normal dropout indicates that dropout is provided to the input gates, while a recurrent dropout implies that dropout is provided to the hidden states across all of the recurrent units.

We bridge a dropout layer between two fully linked layers. Both the first and second completely linked layers employ activation functions called ReLU and Sigmoid, respectively. The loss function is a crucial component of many ML and DL algorithms. The widely used loss function in binary classification issues, Binary Cross-Entropy loss, was the one I utilised for my study. Optimization algorithm is another method that has a major effect on the fields of AI. The Adam optimizer method was employed in this undertaking (Kingma et al., 2017).

We fit the model with a mini-batch size of 32 and over ten epochs. We also take 10% of our training data out as a validation set. This validation set is drawn before model training. The reason behind this is that we want to control the splitting with a set random seed so that our result is somewhat reproducible. We also introduce early stopping on validation loss, so that training is stopped after the epoch that it detects the model is overfitting the training set.

The model summary can be seen in below figure -35 ,

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 100)]             0

 embedding (Embedding)       (None, 100, 128)          12800128

 bidirectional (Bidirectiona  (None, 64)               41216
 l)

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 128)               8320

 dropout_1 (Dropout)         (None, 128)               0

 dense_1 (Dense)             (None, 6)                 774

=================================================================
Total params: 12,850,438
Trainable params: 12,850,438
Non-trainable params: 0
_________________________________________________________________
```

3.4.2.2 LSTM with TextVectorization

Because I was interested in refining my existing baseline model through the use of TextVectorization feature engineering experiments. The model is almost unchanged from my changes.

With 32 vectors making up the vocabulary's 100001 words, the size of the input layer is 100001 * 32. This model is bidirectional model as well however in this model I have manually inserted forward and backword layer with ‘tanh’ activation due to I am using google colab for the implementational of neural network and tanh activation supports gpu very nicely. Since the forward and backward layers has 16 in it consequently bidirectional layer will have 32 nodes in total. After that I have made 3 dense layers with 128,256,128 cells with relu activation.

Since our output will consist of six binary classifications, I included a Sigmoid in the final layer so that the resulting probabilities will range from zero to one. Finally, the model is compiled with the help of the Adam optimizer, the binary cross-entropy loss, and the Accuracy assessment measure. Our model fitting was performed with a mini-batch size of 24, and it took place throughout 10 iterations. We also remove 10% of our training data out as a validation set. This validation set is drawn before model training. The rationale for this is because we want to ensure some degree of reproducibility in our results by regulating the splitting using a known random seed. The model summary may be seen in following f

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, None, 32)          32000032

 bidirectional_1 (Bidirectio  (None, 32)               6272
 nal)

 dense_2 (Dense)             (None, 128)               4224

 dense_3 (Dense)             (None, 256)               33024

 dense_4 (Dense)             (None, 128)               32896

 dense_5 (Dense)             (None, 6)                 774

=================================================================
Total params: 32,077,222
Trainable params: 32,077,222
Non-trainable params: 0
_________________________________________________________________
```

Figure – 36, LSTM Bidirectional Model

4 Results and Evaluation

4.1 Accuracy:

The first (accuracy) is a confusion-matrix-based metric, whose formalisation is given by Equation 1: where t p, t n, f p, and f n stand for true positives, true negatives, false positives, and false negatives, respectively.

accuracy = t p+tn / t p+tn+ f p+ f n

Results:

	Model	Pre-processing	Accuracy	Validation accuracy	Loss
1	Logistic Regression	TF - IDF	0.98	-	-
2	Bidirectional LSTM baseline	Tokenization	0.96	0.99	0.149
3	Bidirectional LSTM	TextVectorization	0.98	0.98	0.0621

Table - 2

Random comment's classification:

Labels = ['toxic', 'severe_toxic', 'obscene', 'threat', 'insult', 'identity_hate']

Comment – ' yo b****h ja rule is more succesful then you ll ever be whats up with you and hating you sad m******s i should b****h slap ur pethedic white faces and get you to kiss my ass you guys sicken me ja rule is about pride in da music man dont diss that shit on him and nothin is wrong bein like tupac he was a brother too f****in white boys get things right next time'

Logistic Regression Prediction – array([1, 0, 0, 0, 1, 0])

Baseline LSTM Prediction– array([1, 0, 1, 0, 1, 0])

LSTM Prediction– array([1, 0, 1, 1, 1, 0])

Additional randomly fetched comment's result of lstm bidirectional model:

```
[93] score_comment('You sucks!!!')

     1/1 [==============================] - 0s 21ms/step
     'toxic: True\nsevere_toxic: False\nobscene: False\nthreat: False\ninsult: True\nidentity_hate: False\n'

[95] score_comment('you are so stupid that you cant even speak!')

     1/1 [==============================] - 0s 22ms/step
     'toxic: True\nsevere_toxic: False\nobscene: False\nthreat: False\ninsult: True\nidentity_hate: False\n'
```

Figure - 37

4.2 Gradio Web application:

This section shows an example of the representation seen in the developed web page.

Sharing machine learning models is made possible by Gradio's collaborative data science workflow creation.

With only a few lines of code, you may build a web-based graphical user interface (GUI) or demo based on a machine learning model (or any Python function) with the help of Gradio's core high-level class, Interface. The user interface function, the input components, and the output components are the three required parameters.

For this project's final deliverable, I developed a gradio-based web app. For the purpose of comment categorization, we are using the best-fitting model, LSTM with TextVectorization. If we feed the programme a bunch of random comments, it should tell us which category each remark belongs to based on its potential toxicity.

Figure – 38, Gradio app.

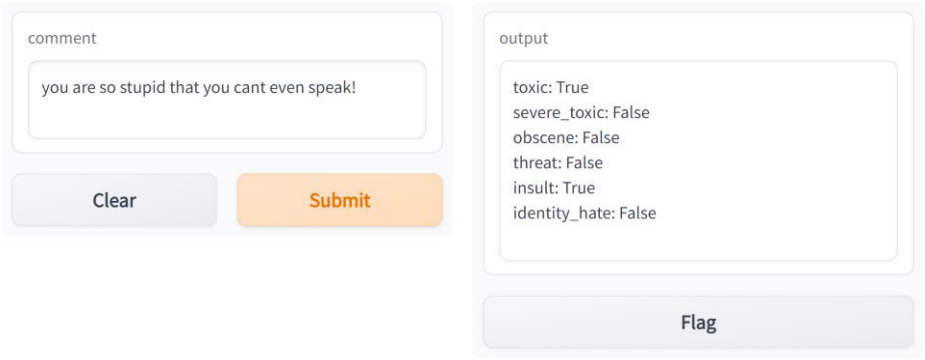
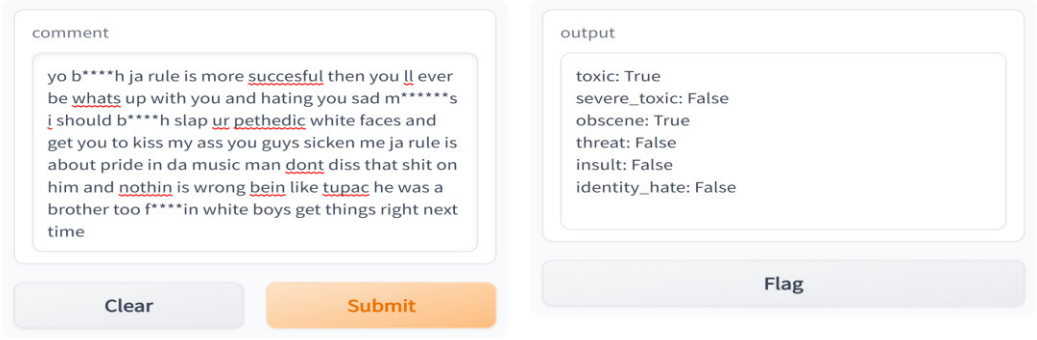


Figure – 39, Gradio App



5 Conclusion and Future Work

One of life's most fundamental need is the ability to communicate with others. The only way for people to share their thoughts is via conversation and interaction. The internet's popularity has skyrocketed over the years, leading to a corresponding growth in the prominence of various forms of media and social networking. Daily, a deluge of data is produced by online communication as individuals talk, express themselves, and voice their views.

The potential for great good to come from this scenario, and for it to improve people's lives, is matched only by the possibility of great harm. It is up to the social media platform to regulate and moderate the remarks made by users.

Numerous social media sites employ some type of content moderation in an effort to eliminate the spread of harmful materials. There has to be fair, scalable tools in place to identify potentially harmful information while it is being posted online. Users will have more faith in these algorithms if they can see that the harmful content classification is tied to a specific section of text. It's important for individuals to have access to social media sites that are devoid of harmful content in order to foster constructive dialogue.

In this study, we want to create a model that, given only a comment's content and tone, can predict whether or not it is toxic using logistic regression. Goal of creating a multi-faceted model for identifying many forms of online toxicity, including but not limited to threats, obscene, insults, and identity hatred. A multi-headed model will be trained on a dataset consisting of toxicity-classified comments using inverse document frequency (TF-IDF) to detect various forms of toxicity, and then evaluated using accuracy measures.

The main research question is the question of the suitability of classifier for classifying toxicity in online comments. For these three different architectures have been developed and evaluated on the Kaggle data set. The results of these experiments show that simpler models do not perform better on this task than deeper ones with more than one layer.

We successfully used LSTM to construct a model to categorise toxic comments and did so with a high degree of accuracy and at a reasonable cost. While the basic model using Tokenization performs well in terms of accuracy, it fails to categorise all the classes when evaluated on random comments. This could be because the model is overfitting, and instead of continuing to run for 10 epochs, it terminated after 4.

However, a bidirectional LSTM model that includes vectorization as well as a forward and a backward layer outperforms both the Logistic and baseline models in terms of accuracy. After putting it through its paces using random remarks, it does an excellent job of classifying test texts.

To enhance the performance of the model, several suggestions can be done. First, since an imbalance of the dataset, we may use several methods and algorithms to balance the data, such as manual annotation, text augmentation, or using several balancing algorithms such as Synthetic Minority Oversampling Technique (SMOTE), WEMOTE/CWEMOTE, Penalize, etc. Subsequently, to enrich the knowledge of the model in recognizing the words input, we can add word embedding such as word2vec, FastText, etc. An interesting future work would be the experimentation of a combination of different contextual word embeddings and deep learning approaches, as well as an ensemble strategy, in order to improve the overall performance.

Finally, it's worth praising the Conversation AI team for their noble goal of creating an open source tool to track and mitigate harmful content online. Many researchers and administrators of online discussion forums have already come up with innovative uses for it. With the machine learning community's support, we can hopefully keep our online dialogues safe from harassment and keep Perspective API running smoothly. If the existing paradigm can be improved, perhaps online debate can become more fruitful and civilised.

6 Reflection

The completion of this thesis marks the completion of several objectives. The first and foremost is an education on the basics of how neural networks function computationally. You now have a solid foundation upon which to build your future abilities and better yourself. The key challenge of this study was sentence categorization, and the LSTM model has been shown to have achieved great accuracy in this area. With respect to pre-processing, it must be mentioned that significantly better results were anticipated to be obtained while training in a pre-processing environment, but this turned out to be erroneous, as the outcomes were essentially the same.

The capacity to retrieve useful information from the neural network cells is the most difficult aim of this thesis and what gives it a unique and individual flavour. The LSTM model is the only one that has had this enhancement applied to it because of how naturally it behaves internally. It has been completed and reworked for better user-friendly presentation in a web app.

With the help of this thesis, I was able to delve into the inner workings of a neural network, learn the logic behind its decisions, and portray that logic graphically. All of this effort is motivated by a desire to demystify a subfield of AI for a general audience by providing straightforward yet technically precise explanations. It may be intriguing to take things a little farther and examine the precise calculations performed by the LSTM, CNN, and GRU. The activations might also be used as an attention layer to boost the model's performance.

7 References

Gupta, K & Muresan, S., 2019. Pay “attention” to your context when classifying abusive language. In Proceedings of the third workshop on abusive language online., pp. 70-79.

Piotr Bojanowski, Edouard Grave, Armand Joulin & Tomas Mikolov, 2017. Enriching word vectors with subword information. Transactions of the Association for, Volume 5, pp. 135-146.

A. A. Sagar & J. S. Kiran, 2008. Toxic comment classification using natural language processing.

A. Y. Taha & S. Tiun, 2016. Binary relevance (br) method classifier of multi-label classification for arabic text. Journal of Theoretical & Applied Information Technology.

Abhigyan, 2020. Understanding Logistic Regression. [Online]
Available at: https://medium.com/analytics-vidhya/understanding-logistic-regression-b3c672deac04
[Accessed 2022].

Adadi, A & Berrada, M., 2018. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). IEEE Access, pp. 6, 52,138–52,160.

Alexandre Ashade Lassance Cunha, Melissa Carvalho Costa & Marco Aur´elio, 2019. Sentiment analysis of youtube video comments using deep neural networks.. In International Conference on Artificial Intelligence and Soft Computing, Springer, pp. 561-570.

Alpaydin, E., 2010. Introduction to Machine Learning. 2 ed. Cambridge: The MIT Press.

Alsaleem, S., 2011. Automated Arabic Text Categorization Using. International Arab Journal of e-Technology, Volume 2.

Barrett, P, et al., 2005. matplotlib--A Portable Python Plotting. In Astronomical data analysis software and systems, Volume 347, p. 91.

Batalla, J. M, et al., 2015. On Providing Cloud-awareness to Client's DASH Application by Using. Journal of Telecommunications and Information Technology.

Chaudhary, M., 2020. TF-IDF Vectorizer scikit-learn. [Online]
Available at: https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a
[Accessed 2022].

Chunting Zhou, Chonglin Sun, Zyiyuan Liu & Francis C.M Lau, 2015. A C-LSTM Neural Network for Text Classification, Hong Kong: Department of Computer Science, The University of Hong Kong.

Colah, 2018. Understanding LSTM Networks - Colah's Blog. [Online]
Available at: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
[Accessed 2022].

Cornegruta, 2016. Modelling Radiological Language with Bidirectional Long Short-Term Memory Networks. s.l.:s.n.

Davidson, T, Warmsley, D, Macy M & Weber, I., 2017. Automated hate speech detection and the problem of offensive language. Proceedings of the ICWSM.

Devin Sony, 2018. Introduction to Naive Bayes Classification. [Online]
Available at: https://towardsdatascience.com/introduction-to-naive-bayes-classification-4cffabb1ae54
[Accessed 2022].

Duggan, M., 2017. Online harrassment.

E. Spishak, W. Dietl & M. D. Ernst., 2012. A type system for regular expressions. FTfJP - In Proceedings of the 14th Workshop on Formal Techniques for Java-like, pp. 20-26.

Erhard Rahm & Hong Hai Do, 2017. Data Cleaning: Problems and Current Approaches. IEEE, pp. 5-10.

fcholle, 2020. Introduction to Keras for Researchers. [Online]
Available at: https://keras.io/getting_started/intro_to_keras_for_researchers/
[Accessed 2022].

Fontaine, A., 2018. Mastering Predictive Analytics with scikit-learn and TensorFlow: Implement machine learning techniques to build advanced predictive models using Python. Packt Publishing.

Fontaine, A., 2018. Mastering Predictive Analytics with scikit-learn and TensorFlow: Implement machine learning techniques to build advanced predictive models using Python. In: s.l.: Packt Publishing.

Gholizadeh, S., 2022. Top Popular Python Libraries in Research. Journal of Robotics and Automation Research, Volume 2, pp. 142-145.

Google Survey, 2019. Be Internet Awesome, s.l.: Google.

Google's Intro to TensorFlow, n.d. *Google's Intr*o to TensorFlow. [Online]
Available at: https://developers.google.com/machine-learning/crash-course/first-stepswith-tensorflow/toolkit.

Gurinder Singh, Bhawna Kumar, Loveleen Gaur & Akriti Tyagi, 2019. Comparison between Multinomial and Bernoulli Naïve Bayes for Text Classification. IEEE, pp. 5-8.

Guven, B., 2020. Data Cleaning in Python using Regular Expressions. [Online]
Available at: https://sonsuzdesign.blog/2020/05/20/data-cleaning-in-python-using-regular-expressions/
[Accessed 2022].

Hovy, D & Waseem, Z., 2016. Hateful symbols or hateful people? Predictive features for hate speech detection on Twitter. In Proceedings of the NAACL student research workshop., pp. 88-93.

Internet Troll, 2019. InternetTroll. [Online].

J. Angwin & H. Grassegger, 2017. *Facebook's secret censorship rules protect white men from hate* speech but not black children.. [Online]
Available at: https://www.propublica.org/article/facebook-hate-speech-censorshipinternal-documents-algorithms
[Accessed 2022].

Jacob, Rabih, Schwartz & Makhoul, 2014. Fast and robust neural network join models for statistical machine translation. Computational Linguistics, p. 1375.

Krishna Dubey, Rahul Nair, Mohd. Usman Khan & Prof. Sanober Shaikh, 2020. Toxic Comment Detection using LSTM. IEEE, pp. 5-10.

Lapidot-Lefler & Barak, A, 2012. Effects of anonymity, invisibility, and lack of eye-contact on toxic online disinhibition. Comput. Hum. Behav., 28(2), pp. 434-443.

Liao S. & Jiang M, 2005. An Improved Method of Feature Selection Based on Concept Attributes in Text Classification. Lecture Notes in Computer Science, pp. 1140-1150.

Liu YY, et al., 2011. A comparison of logistic regression, classification and regression tree, and neural networks models in predicting violent re-offending. J Quant Criminol, Volume 27, pp. 547-560.

Malmasi, S & Zampieri, M, 2017. Detecting Hate Speech in Social Media. Proceedings of the International Conference on Recent Advances in Natural Language Processing, pp. 467-472.

MathWorks, 2019. LSTM Applications and Examples. [Online]
Available at: https://uk.mathworks.com/discovery/lstm.html?s_tid=srchtitle_LSTM_1
[Accessed 2022].

Mathworks, 2019. What Is Deep Learning?. [Online]
Available at: https://uk.mathworks.com/discovery/deep-learning.html
[Accessed 2022].

Maya Shwayder & Mythili Sampathkumar , 2020. Cyberbullying increases amid coronavirus *pandemic. Here's what parents can do.* [Online]
Available at: https://www.digitaltrends.com/news/coronavirus-cyberbullying-distance-learning/

Mike Schuste & Kuldip K. Paliwa, 1997. Bidirectional recurrent neural networks. In:Signal Processing, IEEE Transactions, p. 50.

Mitchell, T., 1997. Machine Learning. s.l.:McGraw Hill.

Mueller, A., 2020. WordCloud for Python documentation. [Online]
Available at: http://amueller.github.io/word_cloud/#
[Accessed 2022].

Mujahed A. Saif, Alexander N. Medvedev, Maxim A. Medvedev & TodorkaAtanasova, 2018. Classification of Online Toxic Comments Using the Logistic Regression and Neural Networks Models. International Advanced Research Journal in Science, Engineering and Technology (IARJSET).

Oliphant, T. E., 2006. A guide to NumPy. USA: Trelgol Publishing, Volume 1, p. 85.

Pandey, K & Panchal, R, 2019. A Study of Real World Data Visualization of COVID-19 dataset using Python. Internation-al Journal of Management and Humanities (IJMH), Volume 4, pp. 1-4.

Pang, B, Nijkamp, E & Wu, Y. N, 2019. Deep learning wit tensorflow. A review. Journal of Educational and Behavioral Statistics, Volume 45, pp. 227-250.

Pedregosa, F, et al., 2012. Scikit-learn: Machine learning in python. the Journal of machine Learning research, Volume 12, pp. 2825-2830.

Pinkesh Badjatiya, Shashank Gupta, Manish Gupta & Vasudeva Varma, 2017. Deep learning for hate speech detection in tweets. In Proceedings of the 26th International Conference on World Wide Web Companion, pp. 759-760.

Preethi, 2018. What is logistic regression?. [Online]
Available at: https://medium.com/@tpreethi19/what-is-logistic-regression-4251709634bb
[Accessed 2022].

raghakot, 2015. karas - text Documentation. [Online]
Available at: https://raghakot.github.io/keras-text/keras_text.processing/
[Accessed 2022].

Revati Sharma & Meetkumar Patel, 2018. Toxic comment classification using Neural networks and machine learning. International Advanced Research Journal in Science, Engineering and Technology (IARJSET), 5(9), pp. 47-52.

Richard Socher, et al., 2013. Recursive deep models for semantic compositionality over. In Proceedings of Empirical Methods on Natural Language Processing, p. 1642.

Sebastiani, F., 2002. Machine learning in automated text categorization. ACM Computing Surveys.

Silaparasetty, N., 2020. The Tensorflow Machine Learning. Berkeley, CA: Apress.

Smith, Peter K, et al., 2008. Cyberbullying: its nature and impact in secondary school pupils. The journal of child psychology and psychaitry, 49(4), pp. 376-385.

Subbu, R. S., 2017. Brief Study of Classification Algorithms in Machine Learning, New York: CUNY City College .

Sweta Karlekar & Mohit Bansal, 2018. Safecity: Understanding diverse forms of sexual. UNC Chapel Hill.

Waskom, M. L., 2021. Seaborn: statistical data visualization. Journal of Open Source Software, p. 6.

Xu J & Wang Z, 2004. An Eficient M ethod of Text Clustering Based on LSI and SNN. Journal of Tianjin University, pp. 1026-1034.

Yaakov HaCohen-Kerner, Daniel Miller, Yair Yigal & Weinan Zhang, 2020. The influence of preprocessing on text classification using a bag-of-words representation. PLoS One, pp. 5-8.

Yang S, 2019. An Introduction to Naïve Bayes Classifier. [Online]
Available at: https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf
[Accessed 2022].